SISTEMAS DIGITALES

TEMA 3:

SISTEMAS ARITMÉTICOS

TEMA 3: SISTEMAS ARITMÉTICOS

INTRODUCCIÓN Y OBJETIVOS

En este capítulo se va a presentar la forma de representación de la información que maneja un computador, haciendo especial énfasis en la representación numérica.

Introduciremos en primer lugar los sistemas numéricos posicionales y dentro de ellos las representaciones en binario puro, octal, hexadecimal y BCD.

Con estos códigos solamente se pueden realizar operaciones aritméticas sin signo, lo cual supone restringir dichas operaciones, a la realización de sumas.

Posteriormente se presentarán los códigos numéricos posicionales, que permitirán realizar operaciones enteras con signo; así veremos las representaciones de signo/magnitud, complemento a 1 y complemento a 2

Para finalizar se introducirán las operaciones algebraicas de suma y resta con diferentes códigos de representación, así como algunas implementaciones de los sistemas físicos que las realizan.

1. REPRESENTACIÓN Y CODIFICACIÓN DE LA INFORMACIÓN (I)

Dada la naturaleza *bimodal* (*biestado*) de los circuitos digitales, la información se representa mediante códigos binarios. En ellos se considera:

1. Elementos a representar (números, cadenas alfanuméricas)
2. Forma de representación (binario puro, coma flotante,….)
3. Características de la representación (con paridad, tolerante a fallos,…..)

1. Representaciones Alfanuméricas (de texto)
2. Representaciones Numéricas

1. REPRESENTACIÓN Y CODIFICACIÓN DE LA INFORMACIÓN (II)

Códigos alfanuméricos. (I)

1. **La representación de las informaciones de tipo texto se realiza codificando, mediante un conjunto de bits, cada uno de los caracteres que componen dicha información.**

2. **Las características que definen un sistema de representación alfanumérico son:**
   - Tamaño de la palabra de datos empleada.
     - De este tamaño depende el número de caracteres distintos representables. (n bits → 2^n caracteres)
   - Codificación de cada carácter.

1. REPRESENTACIÓN Y CODIFICACIÓN DE LA INFORMACIÓN (III)

Códigos alfanuméricos. (II)

3. **Podemos resaltar los siguientes:**

   1. ASCII (American Standard Code for Information Interchange), se representa con 7 bits 128 caracteres diferentes, pensado para el idioma inglés
   - **Códigos del 0 al 31**: caracteres de control de periféricos (no imprimibles).
   - **Códigos del 48 al 57**: dígitos del 0 al 9.
   - **Códigos del 65 al 90**: letras mayúsculas (A- Z).
   - **Códigos del 97 al 122**: letras minúsculas (a- z).
   - **Resto de códigos hasta el 127**: signos diversos. Por ejemplo: ! " # $ % & ' ( ) * + , - . / : ;

1. REPRESENTACIÓN Y CODIFICACIÓN DE LA INFORMACIÓN (IV)

Códigos alfanuméricos (III)

2. ISO-8859-1 es un código ASCII extendido, representa en 8 bits los mismos 128 primeros caracteres definidos en el código ASCII y añade otros 128 no estándar, incluye lenguas de europa occidental y otras

- La letra Ñ.
- Letras acentuadas.
- Letras griegas.
- Caracteres semigráficos. Ejemplos: ® ¼ ÷

3. ISO-8859-15 Modificación del anterior para incorporar nuevos símbolos (euro y otras actualizaciones)
4. ISO-8859-2 a ISO-8859-14 son extensiones del ASCII para diferentes lenguas (cirílico, árabe, hebreo, griego,…)

2. SISTEMAS NUMÉRICOS POSICIONALES (I)

1. **En un sistema posicional, un número viene definido por una cadena de dígitos.**

2. **Cada dígito está afectado por un factor de escala, que depende de la posición (lugar) que ocupa dentro de la cadena.**

3. **El sistema de numeración posicional clásico es el decimal.**
   - En éste, cada dígito de la cadena indica la potencia de 10 que le afecta.
   - Ejemplo: $562_{(10} = 5 \times 10^2 + 6 \times 10^1 + 2 \times 10^0 = 500+60+2$

2. SISTEMAS NUMÉRICOS POSICIONALES (II)

4. **Formalmente:**

Un número *X*, representado por una cadena de dígitos:

$$X \equiv (\ldots\ x_3\ x_2\ x_1\ x_0\ x_{-1}\ x_{-2}\ x_{-3}\ \ \ldots)$$

a través de un vector de pesos seleccionados del conjunto:

$$P \equiv (\ldots\ p_3\ p_2\ p_1\ p_0\ p_{-1}\ p_{-2}\ p_{-3}\ \ \ldots)$$

permite calcular el valor V(X) del número X, mediante la regla:

$$V(X) = \ldots\ p_3 \cdot x_3 + p_2 \cdot x_2 + p_1 \cdot x_1 + p_0 \cdot x_0 + p_{-1} \cdot x_{-1} + p_{-2} \cdot x_{-2} + p_{-3} \cdot x_{-3} + \ldots$$

2. SISTEMAS NUMÉRICOS POSICIONALES (III)

- En los sistemas posicionales lo común es utilizar una base b (radical b) para generar el vector de pesos mediante sus potencias sucesivas, de forma que:

$$P \equiv (\ldots\ b^3\ b^2\ b^1\ b^0\ b^{-1}\ b^{-2}\ b^{-3}\ \ldots)$$

- En este caso ya no es necesario indicar el vector de pesos, tan solo la base empleada.
- Lo habitual es emplear para la base números naturales.

2. SISTEMAS NUMÉRICOS POSICIONALES (IV)

$$V(x) = \sum_{i=m-1}^{n-1} b^i x_i$$

Siendo:

- V(x)= valor del número (magnitud)
- b = base
- X_i = dígito de la cadena
  - x_{n-1}= más significativo
  - x_{m-1}= menos significativo
- n = nº de cifras de la parte entera
- m = nº de cifras de la parte fraccionaria

$V(X)= x_{n-1}b^{n-1}+b^{n-2}\cdot x_{n-2}+\cdots+x_1b^1+x_0b^0\cdot+x_{-1}b^{-1}+x_{-2}b^{-2}\cdots+x_{-m}b^{-m}$

Radicales usuales son:

- b= 10 (sistema <u>decimal</u>).
- b= 2 (sistema <u>binario</u>).
- b= 8 (sistema <u>octal</u>).
- b= 16 (sistema <u>hexadecimal</u>).

2. SISTEMAS NUMÉRICOS POSICIONALES (V)

- También <u>es habitual</u> la imposición de la restricción de que todos los dígitos **x_i** de la cadena sean positivos y menores que la base:
  - b= 10 (sistema decimal):
    x_i= { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }
  - b= 2 (sistema <u>binario</u>):
    x_i= { 0, 1 }
  - b= 8 (sistema <u>octal</u>):
    x_i= { 0, 1, 2, 3, 4, 5, 6, 7 }.
  - b= 16 (sistema <u>hexadecimal</u>):
    x_i= { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F }.

2. SISTEMAS NUMÉRICOS POSICIONALES (VI)

Características de los sistemas de Representación

- RANGO de un sistema de representación es el intervalo comprendido entre el menor y mayor números representables.
- RESOLUCIÓN de un sistema de representación es la diferencia en magnitud entre dos números consecutivos representables.
- DESBORDAMIENTO El número de bits de la palabra es insuficiente para la representación de la magnitud del número.

  Invasión del bit del signo en representaciones con signo

REPRESENTACIÓN EN BINARIO NATURAL (I)

Un número *X* representado por una cadena de dígitos de longitud *n*:

$$(x_{n-1}\ x_{n-2}\ \ldots\ x_2\ x_1\ x_0)$$

tiene por valor:

$$V(X)= 2^{n-1} \cdot x_{n-1} + \ \ldots\ + 2^2 \cdot x_2 + 2^1 \cdot x_1 + 2^0 \cdot x_0$$

Siendo x_i = { 0, 1 }

Es un sistema posicional con base igual a 2 y sin parte fraccionaria.

REPRESENTACIÓN EN BINARIO NATURAL (II)

- **Ejemplos para n= 3:**
  - 001= 1_{10}
  - 110= 6_{10}
  - 111= 7_{10}
- **Ejemplos para n= 12:**
  - 000000000110= 6_{10}
  - 100000000000= $2^{11}{}_{10}$= 2048_{10}
  - 111111111111= $(2^{11}+ 2^{10}+ \ldots + 2^{1}+ 2^{0})_{10}$= $(2^{12}-1)_{10}$= 4095_{10}

REPRESENTACIÓN EN BINARIO NATURAL (III)

Características:

- Rango: **[0, 2^n-1]**, siendo n el número de dígitos binarios o bits (**bi**nary digi**ts**).
- Resolución: **1**.
- En la suma de 2 cantidades, se presentará un problema de desbordamiento si se produce acarreo en el último bit; es decir, si c_{n-1}= **1**.
- Se podrá efectuar la resta de A-B sólo si se cumple que $A \geq B$.

REPRESENTACIÓN EN OCTAL (I)

Un número *X* representado por una cadena de dígitos de longitud *n*:

$$(x_{n-1}\ x_{n-2}\ \ldots\ x_2\ x_1\ x_0)$$

tiene por valor:

$$V(X)= 8^{n-1}\cdot x_{n-1}+\ \ldots\ + 8^2\cdot x_2+ 8^1\cdot x_1+ 8^0\cdot x_0$$

Siendo x_i = { 0, 1, 2, 3, 4, 5, 6, 7,}

Es un sistema posicional con base igual a 8 y sin parte fraccionaria.

REPRESENTACIÓN EN OCTAL (II)

- Ejemplos para n= 3:
  - $172_8 = 122_{10}$
  - $43_8 = 35_{10}$
  - $121_8 = 81_{10}$
- Ejemplos para n= 8:
  - $00000172_8 = 122_{10}$
  - $00000043_8 = 35_{10}$
  - $00000340_8 = (0.8^7 + 0.8^6 + \ldots + 3.8^2 + 4.8^1 + 0.8^0)_{10} = 224_{10}$

REPRESENTACIÓN EN HEXADECIMAL (I)

Un número *X* representado por una cadena de dígitos de longitud *n*:

$$(x_{n-1}\ x_{n-2}\ \dots\ x_2\ x_1\ x_0)$$

tiene por valor:

$$V(X) = 16^{n-1}\cdot x_{n-1} + \dots + 16^2\cdot x_2 + 16^1\cdot x_1 + 16^0\cdot x_0$$

Siendo x_i = { 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}

Es un sistema posicional con base igual a 16 y sin parte fraccionaria.

REPRESENTACIÓN EN HEXADECIMAL (II)

- Ejemplos para n= 3:
  - $1A2_H$
  - 43_H
  - $121_H = 81_{10}$
- Ejemplos para n= 8:
  - 00000172_H
  - $0000A043_H$
  - $000001A0_8 = (0.16^7 + 0.16^6 + \dots + 1.16^2 + A.16^1 + 0.8^0)_{10} = 416_{10}$

REPRESENTACIÓN EN DECIMAL CODIFICADO BINARIO (BCD) (I)

En el sistema decimal codificado en binario o *BCD* (*Binary Coded Decimal*), se convierten los dígitos decimales, uno a uno a binario.

- Las representaciones alfanuméricas, como la definida a través del código ASCII son representaciones BCD que emplean 7 bits por dígito (8 para ASCII extendido).

Dado que hay más de un código de este tipo, el más habitual codifica cada dígito decimal empleando una representación en binario puro.

- Para poder representar un dígito decimal se requieren al menos 4 bits.

REPRESENTACIÓN EN DECIMAL CODIFICADO BINARIO (BCD) (II)

CÓDIGO	VALOR
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
0001 0000	10

CONVERSIONES ENTRE REPRESENTACIONES NUMÉRICAS (I)

Cambio de base: de base 10 a base b

Parte entera: dividiendo sucesivamente por la base y tomando el ultimo cociente y los restos desde el último al primero, hasta que se produzca un cociente menor o igual al divisor .

Parte decimal: multiplicando sucesivamente por la base y tomando la parte entera de cada producto, desde la primera a la última, hasta que obtener solamente parte entera.

Ejemplo: de decimal a octal: $124{,}390625_{(10}$

124 | 8
4   15 | 8
7   1

$0{,}390625 \times 8 = 3{,}125$ (3)
$0{,}125 \times 8 = 1$ (1)

$124{,}390625_{(10} = 174{,}31_{(8}$

CONVERSIONES ENTRE REPRESENTACIONES NUMÉRICAS (II)

Conversión decimal a binario.

- El proceso de obtención se realiza efectuando divisiones sucesivas, entre 2, del número expresado en base 10.
- Condición de parada del algoritmo:
  - Cuando el resultado en una división produzca un cociente menor o igual al divisor (en este caso 2).
  - Este cociente será el dígito de mayor peso del resultado.
- Los restos parciales de las divisiones anteriores, tomados en orden inverso, serán los valores que multipliquen a las potencias de la base.

CONVERSIONES ENTRE REPRESENTACIONES NUMÉRICAS (III)

Ejemplo de conversión decimal a binario.

$124_{10} = 1111100_2$

```
124 | 2
 0    62 | 2
       0   31 | 2
            1   15 | 2
                 1   7 | 2
                     1   3 | 2
                         1   1
```


CONVERSIONES ENTRE REPRESENTACIONES NUMÉRICAS (IV)

Conversión decimal a octal.

- El mismo procedimiento, que de decimal a binario, aunque ahora se realiza dividiendo sucesivamente entre 8.

Ejemplo de conversión decimal a octal.

- $124_{10} = 174_8$

```
124 | 8
 4    15 | 8
       7   1
```

CONVERSIONES ENTRE REPRESENTACIONES NUMÉRICAS (V)

Conversión decimal a hexadecimal.

- El mismo procedimiento, que de decimal a binario, aunque ahora se realiza dividiendo sucesivamente entre 16.
- Para restos comprendidos entre 10 y 15 habrá que sustituirlos por el dígito hexadecimal asociado:
  - $10 \rightarrow A$
  - $11 \rightarrow B$
  - $12 \rightarrow C$
  - $13 \rightarrow D$
  - $14 \rightarrow E$
  - $15 \rightarrow F$

CONVERSIONES ENTRE REPRESENTACIONES NUMÉRICAS (VI)

Ejemplo de conversión decimal a hexadecimal.

- $124_{10} = 7C_{16}$

$$\begin{array}{r|l} 124 & 16 \\ \hline \mathit{12} & \mathit{7} \end{array}$$

Recordando que:

$12 \rightarrow C$

CONVERSIONES ENTRE REPRESENTACIONES NUMÉRICAS (VII)

Conversión de código binario a octal: agrupando dígitos de 3 en 3.

- Empezando por la izquierda de la coma decimal en el análisis de la parte entera, (se pueden añadir ceros no significativos)
- Empezando por la derecha de la coma decimal para la parte decimal. (se pueden añadir ceros no significativos)

Conversión de código binario a hexadecimal: agrupando dígitos de 4 en 4.

- El mismo procedimiento que para el caso de binario a octal.

- Ejemplo:
  - $100111100110{,}0011_2 = 100111100110{,}001100_2 = 4746{,}14_8 = 9E6{,}3_{16}$

CONVERSIONES ENTRE REPRESENTACIONES NUMÉRICAS (VIII)

Conversión binario a octal.

- Empezando por el bit menos significativo de la parte entera se agrupan los dígitos en conjuntos de 3 bits.
- Cada grupo de 3 bits se asocia con el dígito octal correspondiente.

Ejemplo de conversión binario a octal.

- **$011101001_2 = 351_8$**
  - **$001_2 = 1_8$**
  - **$101_2 = 5_8$**
  - **$011_2 = 3_8$**

CONVERSIONES ENTRE REPRESENTACIONES NUMÉRICAS (IX)

Conversión octal a binario.

- Cada dígito octal es convertido a un grupo de 3 bits.
- Este grupo de 3 bits se corresponde con el valor binario de dicho dígito.

Ejemplo de conversión octal a binario.

- $174_8 = 001111100_2$
  - $4_8 = 100_2$
  - $7_8 = 111_2$
  - $1_8 = 001_2$

CONVERSIONES ENTRE REPRESENTACIONES NUMÉRICAS (X)

Conversión binario a hexadecimal.

- Empezando por el bit menos significativo de la parte entera, se agrupan los dígitos en conjuntos de 4 bits.
- Cada grupo de 4 bits se asocia con el dígito hexadecimal correspondiente.

Ejemplo de conversión binario a hexadecimal.

- $101011010001_2 = AD1_{16}$
  - $0001_2 = 1_{16}$
  - $1101_2 = D_{16}$
  - $1010_2 = A_{16}$

CONVERSIONES ENTRE REPRESENTACIONES NUMÉRICAS (XI)

Conversión hexadecimal a binario.

- Cada dígito hexadecimal es convertido a un grupo de 4 bits.
- Este grupo de 4 bits se corresponde con el valor binario de dicho dígito.

Ejemplo de conversión hexadecimal a binario.

- $8F_{16}= 10001111_2$
  - $F_{16}= 1111_2$
  - $8_{16}= 1000_2$

3. NUMÉROS ENTEROS CON SIGNO (I)

Este tipo de aritmética considera el signo de los elementos con los que se opera, por tanto se requiere algún convenio para representarlo.

Al realizar operaciones aritméticas, las representaciones han de ser consecuentes → los resultados han de estar representados en el mismo criterio que los operandos.

3. NUMÉROS ENTEROS CON SIGNO (II)

1. Coma fija y signo magnitud

2. Complemento restringido a la base o complemento a 1, C1

3. Complemento a la base o complemento a 2, C2

3.1 SIGNO-MAGNITUD (I)

En este convenio de representación, un número X está representado por una cadena de dígitos de longitud *n*:

$(x_{n-1}\ x_{n-2}\ \ldots\ x_2\ x_1\ x_0)$

y tiene por valor:

Si $x_{n-1}=0$ $\rightarrow V(X)= +(2^{n-2}\cdot x_{n-2}+ \ldots + 2^2\cdot x_2+ 2^1\cdot x_1+ 2^0\cdot x_0)=$

$$= +\sum_{i=0}^{n-2} 2^i x_i$$

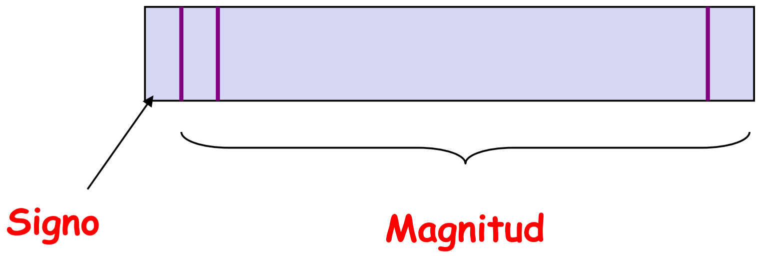
Si $x_{n-1}=1$ $\rightarrow V(X)= -(2^{n-2}\cdot x_{n-2} + \ldots + 2^2\cdot x_2+ 2^1\cdot x_1+ 2^0\cdot x_0)=$

$$= -\left(\sum_{i=0}^{n-2} 2^i x_i\right)$$

3.1 SIGNO-MAGNITUD (II)

Del valor del número puede deducirse, que el bit más significativo x_{n-1} “determina” si el número es positivo o negativo.

Se reserva el bit de la izquierda de la palabra para el signo; el resto de los bits representa la magnitud del número en binario puro”



3.1 SIGNO-MAGNITUD (III)

1. Ejemplos para n = 3:

$$V(X)= +(2^{3-2}\cdot x_{3-2}+ 2^{0}\cdot x_{0})$$

$$V(X) = -(2^{3-2}\cdot x_{3-2} + 2^{0}\cdot x_{0})$$

$001 = +\ (2^{3-2}\cdot 0 + 2^{0}\cdot 1) = +1_{10}$
$111 = -\ (2^{3-2}\cdot 1 + 2^{0}\cdot 1) = -3_{10}$
$000 = +\ 0_{10}$
$100 = -\ 0_{10}$

2. Ejemplos para n = 8:

$00000011 = +\ 3_{10}$

$10000011 = -\ 3_{10}$

$00000000 = +\ 0_{10}$

$10000000 = -\ 0_{10}$

3.1 SIGNO-MAGNITUD (IV)

Rango de representación: $[-(2^{n-1}-1), 2^{n-1}-1]$.

Para n = 3 bits

Nos positivos 000,.........,011 — +0 ....+3 — RANGO $[0 \ , \ 2^{3-1}-1]$

Nos negativos 111,........., 100 — -3 ....-0 — $[-(2^{3-1}-1) \ , \ -0]$

Resolución: 1

3.1 SIGNO-MAGNITUD (V)

Suma de dos números positivos

A=5 B=3 n= 5 bits → (+5) + (+3)

$5_{(10}=00101_{(S/M}$ $3_{(10}=00011_{(S/M}$

```
  111   ← acarreo
00101
00011
-----
01000(S/M
```

$01000_{(S/M} \rightarrow +(1\cdot 2^3+ 0\cdot 2^2 + 0\cdot 2^1 +0\cdot 2^0) = +8_{(10}$

3.1 SIGNO-MAGNITUD (VI)

Suma de un número positivo y otro negativo, siendo el valor absoluto del positivo mayor que el del negativo. Mediante una <u>resta</u>

A=5 B=3 n= 5 bits → (+5)+(-3)= (+5)-(+3)

$5_{(10} = 00101_{(S/M}$ $3_{(10} = 00011_{(S/M}$

```
00101
   1      acarreo
00011-
-----
00010(S/M
```

$00010_{(S/M} \rightarrow +(0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0) = +2_{(10}$

Si se calculara mediante una <u>suma</u>:

```
  111    acarreo
 00101
 10011+
 -----
 11000(S/M
```

$11000_{(S/M} \rightarrow -(1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0) = -8_{(10}$

No es un resultado válido

3.1 SIGNO-MAGNITUD (VII)

Suma de un número positivo y otro negativo, siendo el valor absoluto del negativo mayor que el del positivo

A=5 B=3 n= 5 bits → (-5)+(+3)= (+3)-(+5)

$5_{10} = 00101_{(S/M}$ $3_{10} = 00011_{(S/M}$

Mediante una <u>resta</u>:

```
 00011
111        adeudo
 00101-
------
111110(S/M
```

$111110_{(S/M} \rightarrow -(1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0) = -30_{(10}$

No es un resultado válido.

Si se calculara mediante una <u>suma</u>:

```
0111       acarreo
00011
10101+
-----
11000(S/M
```

$11000_{(S/M} \rightarrow -(1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0) = -8_{(10}$ Resultado no válido

<u>CONCLUSIÓN</u>: hay que restar del de mayor magnitud el de menor y asignar el signo al resultado

3.1 SIGNO-MAGNITUD (VIII)

Suma de dos números negativos

A=-5 B=-3 n= 5 bits → (-5)+(-3)

$-5_{(10}=10101_{(S/M}$ $3_{(10}=10011_{(S/M}$

```
   111        acarreo
  10011
  10101+
 101000(S/M
```

$101000_{(S/M} \rightarrow +(1\cdot2^3+0\cdot2^2+0\cdot2^1+0\cdot2^0)= +8_{(10}$

desbordamiento del signo

signo +

El resultado debería ser: $-8_{(10}= 11000_{(S/M}$

CONCLUSIÓN: En signo magnitud los bits del signo no se operan con los del resto de la palabra, se asignan al resultado.

3.1 SIGNO-MAGNITUD (IX)

Resumen de las características:

- **El bit de signo se trata de forma diferenciada respecto a los bits de la magnitud.**
- **Rango**: $[-(2^{n-1}-1), 2^{n-1}-1]$.
- **Resolución**: 1.
- **El valor cero tiene dos representaciones**: **000...0** y **100...0**, lo que acarrea dificultades en su tratamiento.
- **Las operaciones de suma y resta** se deciden por el signo de los operandos, lo que conlleva ciertas dificultades. Para **las operaciones de multiplicación y división** no existe este problema.

3.2 COMPLEMENTO A 1 (I)

Un número X está representado por una cadena de dígitos de longitud n:

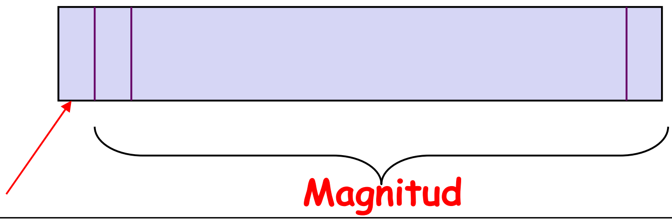
$$(x_{n-1}\ x_{n-2}\ \ldots\ x_2\ x_1\ x_0)$$

y tiene por valor:

Si $x_{n-1}=0 \rightarrow V(X)= 2^{n-2}\cdot x_{n-2}+\ \ldots\ + 2^2\cdot x_2+ 2^1\cdot x_1+ 2^0\cdot x_0 = +\sum_{i=0}^{n-2} 2^i x_i$

Si $x_{n-1}=1 \rightarrow V(X)= 2^{n-2}\cdot x_{n-2}+\ \ldots\ + 2^2\cdot x_2+ 2^1\cdot x_1+ 2^0\cdot x_0 -(2^{n-1}- 1)=$

$$=-(2^{n-1}- 1) + \sum_{i=0}^{n-2} 2^i x_i$$



3.2 COMPLEMENTO A 1 (II)

Sea un número A, representado con n bits. Su valor complementado (complemento a uno del número) es igual a:

$C1(A)= 2^n-1-A$ **(definición)**

Si el número A está codificado en complemento a uno se cumplirá que:

- Si un número es positivo:
  - El bit de mayor peso de la cadena de dígitos($x_{n-1}\ x_{n-2}\ldots x_2\ x_1\ x_0$), que representa al número **será 0**, es decir $x_{n-1}= 0$
  - Por tanto, esta representación coincidirá con la representación en binario puro.
- Si un número es negativo:
  - El bit de mayor peso de la cadena de dígitos, que representa al número **será 1**, es decir, $x_{n-1}= 1$

3.2 COMPLEMENTO A 1 (III)

Una forma muy sencilla y útil de calcular la representación en C1 del número A, es negar uno a uno todos los bits de dicho número, representado este en binario natural.

El complemento a uno del complemento a 1 del número A, es igual a dicho número:

$$C1(C1(A)) = 2^n-1-C1(A) = 2^n-1-(2^n-1-A) = A$$

El cambio de signo del número A puede realizarse simplemente calculando el complemento a uno de dicho número.

3.2 COMPLEMENTO A 1 (IV)

Rango de representación: $[-(2^{n-1}-1), 2^{n-1}-1]$.

Para n = 3 bits

		RANGO
Nos positivos	000,............011 +0 ....+3	$[0 \quad 2^{3-1}-1]$
Nos negativos	100,............111 -3,............-0	$[-(2^{3-1}-1) \quad -0]$

$C1(0) = 2^3-1-0 \Rightarrow 1000 - 1 - 0 = 1111$
$C1(1) = 2^3-1-1 \Rightarrow 1000 - 1 - 1 = 0110$
$C1(3) = 2^3-1-3 = 1000 - 100 = 0100$

Resolución: 1

3.2 COMPLEMENTO A 1 (V)

EJEMPLOS.- Sea A = $11111100_{(C1}$ n=8.

1. Aplicando la definición de la operación de complemento a uno:

$C1(A)= 2^n-1-A$

1	0	0	0	0	0	0	0	0	
1	1	1	1	1	1	1	1	1	-
0	1	1	1	1	1	1	1	1	
	1	1	1	1	1	1	0	0	-
	0	0	0	0	0	0	1	1	

su valor será:

$x_{n-1}=1 \rightarrow V(X)= 2^{n-2}\cdot x_{n-2}+ \ldots + 2^2\cdot x_2+ 2^1\cdot x_1+ 2^0\cdot x_0 -(2^{n-1}- 1)$

$C1(A)= - (2^{8-1}-1) + (1\cdot 2^6+ 1\cdot 2^5+ 1\cdot 2^4+ 1\cdot 2^3+ 1\cdot 2^2+ 0\cdot 2^1 + 0\cdot 2^0) =$
$= - (128-1) + (64 + 32 + 16 + 8 + 4 + 0 + 0) = -127+124 =-3$

2. Este resultado se puede obtener de manera más sencilla negando todos los bits del número.

3.2 COMPLEMENTO A 1 (VI)

EJEMPLOS.- Sea A = $-3_{(10}$ n=8.

1. Representación en C1 de A:
   - A= 11111100

     $V(A) = 1\cdot 2^6+ 1\cdot 2^5+ 1\cdot 2^4+ 1\cdot 2^3+ 1\cdot 2^2+ 0\cdot 2^1+ 0\cdot 2^0- (2^{8-1}-1)$
     $= 124 - (128-1)= 124-127 = -3$
2. Cálculo del valor negativo -A:
   - - A = 3 ➔ C1(A) = 00000011

     (negación bit a bit de la representación de A)
3. Cálculo de -(-A):
   - -(-A)= -(3) = C1(C1(A))= C1(00000011)= 11111100 = A

3.2 COMPLEMENTO A 1 (VII)

EJEMPLOS.-

- A = 01100110

  $V(A) = 1.2^6 + 1.2^5 + 1.2^2 + 1.2^1 = 64+32+4+2 = +102$

- 11100110

  $V(A) = 1\cdot 2^6 + 1\cdot 2^5 + 1\cdot 2^2 + 1\cdot 2^1 - (2^{8-1}-1) = 64+32+4+2-(128-1) = -25$

  $11100110 \longrightarrow 00011001 \longrightarrow 1.2^4 + 1.2^3 + 1.2^0 = +25$

- $00000000 \longrightarrow 0$

- $11111111 \longrightarrow C1(A) = 2^n - 1 - A = [100000000]-1-[11111111] =$

  $=[11111111] - [11111111] = 00000000$

- $[-(2^{n-1}-1), +(2^{n-1}-1)]$ para $n=8 \longrightarrow [-2^7-1, +2^7-1] = [-127, 127]$

3.2 COMPLEMENTO A 1 (VIII)

Resumen de las características:

- El complemento a 1 se reduce a efectuar el complemento lógico (función NOT) de todos los bits.
- El cero tiene 2 representaciones: **00...0** y **11...1**.
- Rango de representación simétrico: **$[-(2^{n-1}-1), +(2^{n-1}-1)]$**.
- En la suma hay que incrementar el resultado en una unidad siempre que el acarreo final sea $c_n = 1$.
- Las operaciones de multiplicación y división se dificultan puesto que los operandos pueden estar complementados.
- Se pueden producir desbordamientos hacia el bit de signo que deben ser detectados.

3.3 COMPLEMENTO A 2 (I)

Un número X representado por una cadena de dígitos de longitud n:

$$(x_{n-1}\ x_{n-2}\ \ldots\ x_2\ x_1\ x_0)$$

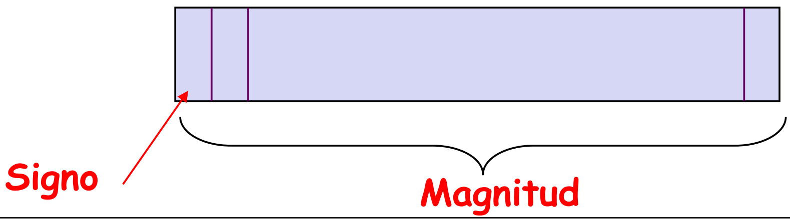
y tiene por valor:

$$\text{Si } x_{n-1}=0 \rightarrow V(X)= 2^{n-2}\cdot x_{n-2}+ \ldots + 2^2\cdot x_2+ 2^1\cdot x_1+ 2^0\cdot x_0=\sum_{i=0}^{n-2} 2^i x_i$$

$$\text{Si } x_{n-1}=1 \rightarrow V(X)= 2^{n-2}\cdot x_{n-2}+ \ldots + 2^2\cdot x_2+ 2^1\cdot x_1+ 2^0\cdot x_0- 2^{n-1}$$

$$= \sum_{i=0}^{n-2} 2^i x_i - 2^{n-1} x_{n-1}$$

Siendo $x_i= \{ 0, 1 \}$



3.3 COMPLEMENTO A 2 (II)

Dado un número A, el complemento a 2 de A, viene dado por:

$C2(A) = 2^n - A$ **(definición)**

$C2(A) = 2^n - A = C1(A) + 1$

Si el número A está codificado en complemento a dos se cumplirá que:

- Si un número es positivo:
  - El bit de mayor peso de la cadena de dígitos$(x_{n-1}\ x_{n-2} \ldots x_2\ x_1\ x_0)$, que representa al número **será 0**, es decir $x_{n-1}= 0$
  - Por tanto, esta representación coincidirá con la representación en binario puro.
- Si un número es negativo:
  - El bit de mayor peso de la cadena de dígitos, que representa al número **será 1**, es decir, $x_{n-1}= 1$

3.3 COMPLEMENTO A 2 (III)

CÁLCULO DEL COMPLEMENTO A 2

* Debemos asegurarnos de que el valor complementado del número de partida (de n bits) puede representarse con el número de bits disponibles.

a) Mediante una resta (definición).

Ejemplo: $A = 7$ y $n = 8$

$A = 00000111$

$-A = 2^8 - 7 = 11111001$

```
  1 0 0 0 0 0 0 0 0
    ❶ ❶ ❶ ❶ ❶ ❶ ❶ ❶
    0 0 0 0 0 1 1 1  -
  -------------------
  0 1 1 1 1 1 0 0 1
```

b) Realizando el Complemento lógico (C1) y sumando 1.

- Ejemplo: A=7 y n=8,

  A= 00000111

  NOT(A)= 11111000

  -A= NOT(A)+1= 11111001

```
  1 1 1 1 1 0 0 0
                1  +
  ---------------
  1 1 1 1 1 0 0 1
```


3.3 COMPLEMENTO A 2 (IV)

Rango de representación: $[-2^{n-1}, 2^{n-1}-1]$.

Para n = 3 bits

N^{os} positivos 000,…………011
+0, ……….+3 $[0 \quad 2^{3-1} -1]$

N^{os} negativos 100,…………111
-4 ……… -1 $[-2^{3-1} \quad -2^0]$

$C2(0) = 2^3 - 0 \Rightarrow 1000 - 0 = 1000$
$C2(1) = 2^3 - 1 \Rightarrow 1000 - 1 = 0111$
$C2(4) = 2^3 - 4 = 1000 - 100 = 0100$

Resolución: 1

3.3. OPERACIONES EN COMPLEMENTO 2 (C2) (V)

1. Cambio de signo.

Consiste en calcular el valor complementado del número.

$-A = C2(A)$

$-(-A) = C2(C2(A)) = C2(2^n - A) = 2^n - (2^n - A) = A$

2. Suma de 2 números negativos.

$(-A)+(-B)= (2^n-A)+(2^n-B)= 2^n-(A+B)+2^n= C2(A+B)+2^n$

Se debe ignorar 2^n. El acarreo de salida se ignora.

Ejemplo: n = 8, A = -3, B = -9

$A= 2^8-3= 11111101$; $B = 2^8-9= 11110111$

$A+B = -12= 2^8-12= 11110100$

		1	1	1	1	1	1	1	
	1	1	1	1	1	1	0	1	
	1	1	1	1	0	1	1	1	+
1	1	1	1	1	0	1	0	0	

3.3. OPERACIONES EN COMPLEMENTO 2 (C2) (VI)

3. Resta (suma algebraica) de 2 números A y B

1. Considerando |A|< |B|.

$A-B= A+(-B)= A+C2(B)= A+(2^n-B)= 2^n-(B-A)= C2(B-A)$

Resultado a la salida es correcto.

Ejemplo: n= 8, A= 3, B= 9

$A= 00000011$; $-B= 2^8-9= 11110111$

$A-B= -6= 2^8-6= 11111010$

				1	1	1		
0	0	0	0	0	0	1	1	
1	1	1	1	0	1	1	1	+
1	1	1	1	1	0	1	0	

2. Considerando $|A| \geq |B|$.

$A-B = A+(-B)= A+(2^n-B)= 2^n+(A-B)$

Se debe ignorar 2^n. El acarreo de salida se ignora.

Ejemplo: n = 8, A = 7, B = 3

$A = 00000111$; $-B = 2^8-3= 11111101$

$A-B = 4= 00000100$

		1	1	1	1	1	1	1	
	0	0	0	0	0	1	1	1	
	1	1	1	1	1	1	0	1	+
1	0	0	0	0	0	1	0	0	

3.3 COMPLEMENTO A 2 (VII)

Resumen de las características:

- Rango de representación asimétrico: $[-2^{n-1}, 2^{n-1}-1]$.
- El valor 0 tiene una única representación: 000...0.
- Las operaciones de suma y resta no tienen que tener en cuenta el signo de los operandos
- Las operaciones de multiplicación y división se complican en el caso de operandos complementados.
- Se pueden producir desbordamientos hacia el bit de signo que deben ser detectados.

3.4 EXTENSIÓN DE SIGNO (I)

Un número X representado con n bits, de la forma:

$X \equiv (x_{n-1}\ x_{n-2}\ \dots\ x_1\ x_0)$

tendrá una representación en m bits, siendo m>n:

$X \equiv (x_{m-1}\ x_{m-2}\ \dots\ x_{n-1}\ x_{n-2}\ \dots\ x_1\ x_0)$

que dependerá del sistema de representación empleado

BINARIO PURO:

Los bits $x_{m-1}\ x_{m-2}\ \dots\ x_n$ serán todos igual a 0.

Ejemplo si n=4 **y** m=5

$0101_{(2} = 00101_{(2} = 5_{(10}$

3.4 EXTENSIÓN DE SIGNO (II)

SIGNO-MAGNITUD

Se conservará el bit de signo x_{n-1}, que pasará a ser el nuevo bit de signo (x_{m-1}). El resto de los bits se rellenará con 0.

$x_{n-1} \rightarrow x_{m-1}$

$0 \rightarrow x_{m-2}$

$0 \rightarrow x_{m-3}$

$\ldots \rightarrow \ldots$

$0 \rightarrow x_{n-1}$

Ejemplo si n=4 y m=5:

$1101_2 = 10101_2 = -5_{10}$

3.4 EXTENSIÓN DE SIGNO (III)

COMPLEMENTO A UNO

Los bits ($x_{m-1}\, x_{m-2} \ldots x_n$) se rellenan con el valor del bit x_{n-1}

$x_{n-1} \rightarrow x_{m-1}$

$x_{n-1} \rightarrow x_{m-2}$

$x_{n-1} \rightarrow x_{m-3}$

$\ldots \rightarrow \ldots$

$x_{n-1} \rightarrow x_n$

Ejemplo si n=4 y m=5

$1101_2 = 11101_2 = -2_{10}$

3.4 EXTENSIÓN DE SIGNO (IV)

COMPLEMENTO A DOS

Los bits ($x_{m-1}\, x_{m-2}\, \ldots\, x_n$) se rellenan con el valor del bit x_{n-1}

$x_{n-1} \rightarrow x_{m-1}$

$x_{n-1} \rightarrow x_{m-2}$

$x_{n-1} \rightarrow x_{m-3}$

$\ldots \rightarrow \ldots$

$x_{n-1} \rightarrow x_n$

Ejemplo si n=4 y m=5

$1101_2 = 11101_2 = -3_{10}$

3.5 RESUMEN: VALOR DE UN NÚMERO (I)

SIGNO/MAGNITUD

Si $x_{n-1}=0 \rightarrow V(X)= +(2^{n-2}\cdot x_{n-2} + \ldots + 2^2\cdot x_2 + 2^1\cdot x_1 + 2^0\cdot x_0)$

Si $x_{n-1}=1 \rightarrow V(X)= -(2^{n-2}\cdot x_{n-2} + \ldots + 2^2\cdot x_2 + 2^1\cdot x_1 + 2^0\cdot x_0)$

Siendo $x_i = \{0, 1\}$

COMPLEMENTO A 1

Si $x_{n-1}=0 \rightarrow V(X)= +(2^{n-2}\cdot x_{n-2} + \ldots + 2^2\cdot x_2 + 2^1\cdot x_1 + 2^0\cdot x_0)$

Si $x_{n-1}=1 \rightarrow V(X)= 2^{n-2}\cdot x_{n-2} + \ldots + 2^2\cdot x_2 + 2^1\cdot x_1 + 2^0\cdot x_0 - (2^{n-1} - 1)$

Siendo $x_i = \{0, 1\}$

COMPLEMENTO A 2

Si $x_{n-1}=0 \rightarrow V(X)= +(2^{n-2}\cdot x_{n-2} + \ldots + 2^2\cdot x_2 + 2^1\cdot x_1 + 2^0\cdot x_0)$

Si $x_{n-1}=1 \rightarrow V(X)= 2^{n-2}\cdot x_{n-2} + \ldots + 2^2\cdot x_2 + 2^1\cdot x_1 + 2^0\cdot x_0 - 2^{n-1}$

Siendo $x_i = \{0, 1\}$

3.5 RESUMEN: VALOR DE UN NÚMERO (II)

Si $x_{n-1}=0$ → N^os positivos $V(X)= +(2^{n-2}\cdot x_{n-2}+ \ldots + 2^2\cdot x_2+ 2^1\cdot x_1+ 2^0\cdot x_0)$

PARA TODAS LAS REPRESENTACIONES

Si $x_{n-1}=1$ → N^os negativos

SIGNO/MAGNITUD

$$V(X)= -(2^{n-2}\cdot x_{n-2} + \ldots + 2^2\cdot x_2+ 2^1\cdot x_1+ 2^0\cdot x_0)$$

COMPLEMENTO A 1

$$V(X)= 2^{n-2}\cdot x_{n-2} + \ldots + 2^2\cdot x_2+ 2^1\cdot x_1+ 2^0\cdot x_0 - (2^{n-1} - 1)$$

COMPLEMENTO A 2

$$V(X)= 2^{n-2}\cdot x_{n-2} + \ldots + 2^2\cdot x_2+ 2^1\cdot x_1+ 2^0\cdot x_0 - 2^{n-1}\, x_{n-1}$$

Siendo en todos los casos $x_i= \{ 0, 1 \}$

3.5 RESUMEN: TABLA DE CÓDIGOS

	CÓDIGO	VALOR EN BINA. NAT.	VALOR EN SIGNO/MAGNITUD	VALOR EN C1	VALOR EN C2
	0000	0	+0	+0	+0
	0001	1	+1	+1	+1
	0010	2	+2	+2	+2
	0011	3	+3	+3	+3
	0100	4	+4	+4	+4
	0101	5	+5	+5	+5
	0110	6	+6	+6	+6
	0111	7	+7	+7	+7
	1000	8	-0	-7	-8
	1001	9	-1	-6	-7
	1010	10	-2	-5	-6
	1011	11	-3	-4	-5
	1100	12	-4	-3	-4
	1101	13	-5	-2	-3
	1110	14	-6	-1	-2
	1111	15	-7	-0	-1
Rango n bits	$(0\ ,\ 2^n -1)$	$(0\ ,\ 2^{n-1}-1)$	$(-(2^{n-1}-1),2^{n-1}-1)$	$(-(2^{n-1}-1),2^{n-1}-1)$	$(-2^{n-1},2^{n-1}-1)$
R (4 bits)	(0 , 15)	(0 , 15)	(-7 , +7)	(-7 , +7)	(-8 , +7)